Method for Encoding and Decoding Free Viewpoint Videos

Field of the Invention

[01] The present invention relates generally to video processing, and more particularly to encoding videos obtained of a moving 3D object by multiple fixed cameras, and decoding and rendering the videos from arbitrary viewpoints.

Background of the Invention

[02] Over the years, telepresence has become increasingly important in many applications including computer supported collaborative work (CSCW) and entertainment.

[03] Such 3D video processing poses a major technical challenge. First, there is the problem of how a 3D video bitstream should be encoded for efficient processing, communications and storage. Second, there is the problem of extracting and reconstructing real moving 3D objects from the videos. Third, it is desired to render the object from arbitrary viewpoints.

[04] Most prior art 3D video bitstreams are formatted to facilitate off-line post-processing and hence have numerous limitations that makes them less practicable for advanced real-time 3D video processing.

[05] Video Acquisition

[06] There are a variety of known methods for reconstructing objects from 2D videos. These can generally be classified as requiring off-line post-processing methods and real-time methods. The post-processing methods can provide point sampled representations, however, not in real-time.

[07] Spatio-temporal coherence for 3D video processing is used by Vedula et al., "Spatio-temporal view interpolation," Proceedings of the Thirteenth Eurographics Workshop on Rendering, pp. 65–76, 2002. There, a 3D scene flow for spatio-temporal view interpolation is computed, however, not in real-time.

[08] A dynamic surfel sampling representation for estimation 3D motion and dynamic appearance is also known. However, that system uses a volumetric reconstruction for a small working volume, again, not in real-time, see Carceroni et al., "Multi-View scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape & reflectance," Proceedings of the 7th International Conference on Computer Vision," pp. 60–67, 2001. Würmlin et al., in "3D video recorder," Proceedings of Pacific Graphics '02, pp. 325–334, 2002, describe a 3D video recorder which stores a spatio-temporal representation in which users can freely navigate.

[09] In contrast to post-processing methods, real-time methods are much more demanding with regard to computational efficiency. Matusik et al., in "Image-based visual hulls," Proceedings of SIGGRAPH 2000, pp. 369–374, 2000, describe an image-based 3D acquisition system which calculates the

visual hull of an object. That method uses epipolar geometry and outputs a view-dependent representation. Their system neither exploits spatio-temporal coherence, nor is it scalable in the number of cameras, see also Matusik et al., "Polyhedral visual hulls for real-time rendering," Proceedings of Twelfth Eurographics Workshop on Rendering, pp. 115–125, 2001.

[010] Triangular texture-mapped mesh representation are also known, as well as the use of trinocular stereo depth maps from overlapping triples of cameras, again mesh based techniques tend to have performance limitations, making them unsuitable for real-time applications. Some of these problems can be mitigated by special-purpose graphic hardware for real-time depth estimation.

[011] Video Standards

[012] As of now, no standard for dynamic, free viewpoint 3D video objects has been defined. Auxiliary components of the MPEG-4 standard can encode depth maps and disparity information. However, those are not complete 3D representations, and shortcomings and artifacts due to DCT encoding, unrelated texture motion fields, and depth or disparity motion fields still need to be resolved. If the acquisition of the video is done at a different location than the rendering, then bandwidth limitations on the transmission channel are a real concern.

[013] Point Sample Rendering

[014] Although point sampled representations are well known, none can efficiently cope with dynamically changing objects or scenes, see any of the following U.S. Patents, 6,509,902, "Texture filtering for surface elements," 6,498,607 "Method for generating graphical object represented as surface elements," 6,480,190 "Graphical objects represented as surface elements," 6,448,968 "Method for rendering graphical objects represented as surface elements," 6,396,496 "Method for modeling graphical objects represented as surface elements," 6,342,886, "Method for interactively modeling graphical objects with linked and unlinked surface elements." That work has been extended to include high-quality interactive rendering using splatting and elliptical weighted average filters. Hardware acceleration can be used, but the pre-processing and set-up still limit performance.

[015] Qsplat is a progressive point sample system for representing and displaying a large object. Qsplat uses a view-dependent progressive transmission technique for a multi-resolution rendering system. Static objects are represented by a multi-resolution hierarchy of point samples based on bounding spheres. Splatting is used to render the point samples. Extensive pre-processing is relied on for splat size and shape estimation, see Rusinkiewicz et al., "QSplat: A multi-resolution point rendering system for large meshes," Proceedings of SIGGRAPH 2000, pp. 343-352, 2000.

[016] Botsch et al. use an octree data structure for storing point sampled geometry, see "Efficient high quality rendering of point sampled geometry," Proceedings of the 13th Eurographics Workshop on Rendering, pp. 53-64,

2002. Typical data sets can be encoded with less than five bits per point for coding tree connectivity and geometry information. When surface normals and color attributes are included, the bit requirements double or triple. A similar compression performance is achieved by a progressive encoding scheme for isosurfaces using an adaptive octree and fine level placement of surface samples, see Lee et al., "Progressive encoding of complex isosurfaces," Proceedings of SIGGRAPH 2003, ACM SIGGRAPH, pp. 471-475, July 2003.

[017] Briceno et al.. in "Geometry videos," Proceedings of ACM Symposium on Computer Animation 2003, July 2003, reorganize data from dynamic 3D objects into 2D images. That representation allows video compression techniques to be applied to animated polygon meshes. However, they cannot deal with unconstrained free viewpoint video as described below.

[018] Vedula et al. in "Spatio-temporal view interpolation," Proceedings of the 13th ACM Eurographics Workshop on Rendering, June 2002, describe a free viewpoint video system based on the computation of a 3D scene flow and spatio-temporal view interpolation. However, they do not address the coding of the 3D scene flow representation.

[019] Another method is described by Wuermlin et al., "3D video fragments: Dynamic point samples for real-time free viewpoint video," Computers & Graphics 28(1), Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data, Elsevier Ltd, 2003, also see U.S. Patent Application 10/624,018, "Differential Stream of Point

Samples for Real-Time 3D Video, filed on July 21, 2003, by Würmlin et al., incorporated herein by reference. That method uses point samples as a generalization of 2D video pixels into 3D space. A point sample holds, additionally to its color, a number of geometrical attributes. The geometrical attributes guarantee a one-to-one relation between 3D point samples and foreground pixels in respective 2D video images. That method does not make any assumptions about the shape of the reconstructed object. The problem with that method is that both the encoder and decoder need to maintain a detailed 3D point model. The progressive sampling used in that system increases the complexity of the system, particularly in the decoder.

[020] None of the prior art methods provide an efficient compression framework for an arbitrary viewpoint video of moving 3D objects.

[021] Therefore, there still is a need for encoding multiple videos acquired of moving 3D objects by fixed cameras, and decoding the encoded bitstream for arbitrary viewpoints.

Summary of the Invention

[022] A system encodes videos acquired of a moving object in a scene by multiple fixed cameras. Camera calibration data of each camera are first determined. The camera calibration data of each camera are associated with the corresponding video.

[023] A segmentation mask for each frame of each video is determined. The segmentation mask identifies only foreground pixels in the frame associated with the object. A shape encoder then encodes the segmentation masks, a position encoder encodes a position of each pixel, and a color encoder encodes a color of each pixel.

[024] The encoded data can be combined into a single bitstream and transferred to a decoder. At the decoder, the bitstream is decoded to an output video having an arbitrary user selected viewpoint.

[025] A dynamic 3D point model defines a geometry of the moving object. Splat sizes and surface normals used during the rendering can be explicitly determined by the encoder, or explicitly by the decoder.

Brief Description of the Drawings

[026] Figure 1 is a block diagram of a system and method for encoding and decoding multiple videos acquired of moving 3D objects according to the invention;

[027] Figure 2 is a block diagram of an encoder used by the system of Figure 1;

[028] Figure 3 is a graph of constrained and unconstrained free viewpoint trajectories;

[029] Figure 4 is a diagram of a scan line traversal of a portion of an image inside a mask;

[030] Figure 5 is a block diagram of an encoder for an unconstrained video;

[031] Figure 6 is block diagram of a base layer and an enhancement layer encoded as a video bitstream according to the invention;

[032] Figure 7A is a key frame according to the invention;

[033] Figures 7B is an images reconstructed from the key frame of Figure 7A.

Detailed Description of the Preferred Embodiment

[034] System Structure

[035] Figure 1 shows the general structure 100 of a system and method for encoding 200 input videos 202 and decoding 400 to output videos 109 according to our invention. Embodiment for encoding constrained and unconstrained free viewpoint videos are shown in greater detail in Figures 2 and 5. Constrained and unconstrained free viewpoint videos according to the invention are described in greater detail below.

[036] As an advantage of our invention, the acquiring can be performed at a local encoding node, and the decoding at a remote decoding node. The

encoding 200 can be performed as an off-line process and is not time-critical. However, the decoding 400 is done in real-time.

[037] We used an encoded video bitstream 208. The bitstream 208 can be transferred from the encoder 200 to the decoder 400 using any conventional means, e.g., a file transfer, an intermediate storage media (DVD), a network channel, etc.

[038] Before acquiring the videos 202, extrinsic and intrinsic parameters of synchronized cameras 101 are estimated. The calibration parameters 209 can include, for example, a projection matrix and a center of a projection vector. The encoder 200 provides the camera calibration data 209 to the decoder 400. The camera calibration data can be sent one time before the video bitstream 208 is transferred as long as the cameras remain fixed. The calibration data 209 can be updated periodically as needed should the camera parameters change due to relocating any of the cameras.

[039] At the encoder, the multiple calibrated and synchronized cameras 101 are arranged around an object 102, e.g., a moving user. Each camera acquires an input sequence of images (input videos) 202 of the moving object 102. For example, we can use fifteen cameras around the object, and one or more above. Other configurations are possible. Each camera has a different viewpoint or 'pose', i.e., location and orientation, with respect to the object 102. This information is encoded in as part of the camera calibration data 209. It is not necessary that all cameras view the object 102 at all times. Indeed, a subset of cameras may sufficient in many cases, for example, only the cameras providing a frontal view of the object 102.

[040] The video processing involves the following steps described in greater detail below. The videos 202 are processed to segment foreground pixels from a background portion in a scene. For this, we use silhouettes or binary segmentation masks 201, see Figure 2. The background portion can be discarded or transferred to the decoder 400 as a single frame to form a virtual scene 151. Alternatively, the virtual scene 151 used during the video reconstruction can be generated synthetically.

[041] It should be noted that the object 102, such as a user, can move relative to the cameras 101. The implication of this is described in greater detail below.

[042] Given an actual rendering viewpoint 401, perhaps selected by a user in real-time, we select 410 a set of active cameras from all of the available cameras 101. The selected cameras have a 'best' view of the user 102 for the actual rendering viewpoint 401. The bitstream 208 is decoded, perhaps at a remote location, using point splatting and the arbitrary camera viewpoint 401. That is, the rendering viewpoint, at any one time, can be different from those of the cameras 101. Interpolation is used to determine pixel values for new arbitrary viewpoints. The interpolation uses images taken by cameras closest to the new viewpoints.

[043] The decoded images 109 can be composited 150 with the virtual scene 151. We can also apply deferred rendering operations, e.g., procedural warping, explosions and beaming, using graphics hardware to maximize

performance and image quality of the reconstructed object in the virtual scene.

[044] Camera Selection at the Decoder

[045] We use camera selection 410 at the decoder 400. The camera selection enables smooth transitions between subsets of cameras, and reduces efficiently the number of cameras required for decoding the 3D bitstream 208. The number of so-called decoding active cameras enables a smooth transition from view-dependent input videos 202 to a view-independent rendering 103 for the output video 109.

[046] For the desired viewpoint 401, we select k cameras that are nearest to the object 102. In order to select the nearest cameras as decoding active cameras, we compare the angles of the viewing direction with the angle of all cameras 101. Selecting the k-closest cameras minimizes artifacts due to occlusions.

[047] The multiple 2D videos 202 acquired by the synchronized cameras 101 provide the inputs to the encoder 200. Additionally, we have at our disposal for every input frame a segmentation mask 202, see Figure 2. The masks indicate the pixels that are part of foreground or object 102.

[048] Each foreground input pixel includes a position, a surface normal vector, a splat size, and color (texture). The simplest way to indicate position is by a depth (z) value. The position, in combination with the camera calibration data 209, define the geometry of a point model 265 of the object

102. Additional data, such as reflectance and refractive information can further describe the visual appearance of the object 102.

[049] Constrained and Unconstrained Videos

[050] For the output video 109, we distinguish between constrained and unconstrained free viewpoint videos. By this we mean, that an arbitrary viewpoint is selected during playback, and the viewpoint can be different than the viewpoints of any of the cameras 101 used to acquire the input videos.

[051] In the constrained free viewpoint video, the point model 265 can be rendered from any possible viewpoint, but the viewpoint (camera) remains constant during the rendering. If discontinuities can be tolerated during the rendering, then changes in viewpoints are allowed as long as the same encoded video can be used.

[052] In an unconstrained free viewpoint video, the point model 265 can be rendered from any possible viewpoint. During playback, the viewpoint is a function of a rendering time. Any discontinuities in rendering due to the changing viewpoints are minimized.

[053] Figure 3 shows examples of spatio-temporal constrained and unconstrained trajectories of viewpoints. In Figure 3, frames 0-M at time t are depicted on the y-axis, and cameras 1-N on the x-axis. In the constrained case, a trajectory 301 lies on a straight or almost straight line corresponding to a single viewpoint, although that viewpoint does not need to match the

viewpoints of any of the cameras 101 used to acquire the input videos. In the unconstrained case, trajectory 302 can be arbitrary in space and time.

[054] Taking into account that the number of cameras is potentially large and that the decoding 400 is in real-time, it is unrealistic on current hardware to first decode all videos from all cameras, and then to render the scene for a given viewpoint. Hence, it is necessary to reduce the number of videos that are decoded.

[055] Thus, view dependent decoding is provided by the method according to the invention, using the following variables and functions:

t	time of recording
θ	time of rendering
$v(\theta)$	viewpoint as a function of the time of rendering,
$D(v(\theta), t)$	set of data decoded for a viewpoint $v(\theta)$, and a time t
$R(v(\theta), t)$	set of data decoded for a viewpoint $v(\theta)$), and a time t, which becomes visible during rendering.

[056] $D(v(\theta), t)$ is the result after decoding and $R(v(\theta), t)$ is the result after rendering. Optimal view dependent decoding is achieved when $D(v(\theta), t) = R(v(\theta), t)$. This implies that the decoder 400, for a given decoded frame only decodes information in the corresponding recorded frame, which becomes visible in the final rendering.

[057] In other words, D is a part of the point model 265 which is decoded, i.e., the contribution of the cameras, which are selected by the camera control at the decoder, and R is the part of the point model 265 which is rendered to an output device, i.e., the part that is visible. In that sense, R is part of an output image. Also, t is the time of recording, which is a discrete time, i.e., the camera frame numbers. The value θ is the time during rendering, i.e., the time when the user selects forward/backward, fast/normal speed playback, and an arbitrary viewpoint.

[058] The strong condition of optimal view-dependent decoding can be relaxed using a weaker formulation for a suboptimal view dependent decoding that maximizes the intersection $D(v(\theta), t) \cap R(v(\theta), t)$. This implies that the decoder, for a given rendering image, maximizes the ratio of the decoded information of the corresponding recorded images, i.e., the decoded information used in the final rendering versus the total amount of decoded information for the given rendering image.

[059] Application Domains

[060] A high compression ratio is efficient when the bitstream 208 is transferred via a low bandwidth network . However, a high compression ratio increases the decoding complexity. In order to support a wide range of target output devices, e.g., cellular telephones, laptops, handheld computers, it is desired to provide a decoder with a relatively low complexity.

[061] The problem of low bandwidth transmissions can be addressed by building upon a progressive representation of the data. In fact, bandwidth

and CPU performance are often correlated, e.g. high-end computing nodes have, in general, access to a high bandwidth network connection, while computing nodes with a low bandwidth network connection generally have limited processing power.

[062] Encoding

[063] Therefore, the format of the bitstream 208 needs to address the following features.

[064] Multi-resolution: Scalability and progressivity with respect to resolution. This can be achieved using either progressive encoding of the data, e.g., embedded zerotree wavelet coding (EZW), see Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on Signal Processing, 41, pp. 3445–3462, December 1993, progressive JPEG, or progressive sampling of the data, as described by Wuermlin et al. in "3D video fragments: Dynamic point samples for real-time free viewpoint video," Computers & Graphics 28(1), Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data, Elsevier Ltd, 2003.

[065] The progressive encoding is preferred over progressive sampling of Wuermlin et al. because of the lower decoding complexity.

[066] Multi-rate: Scalability with respect to time, i.e., the playback of the output video 109 is possible at a different frame rate than the frame rate used

for recording the input videos 202. Backward (reverse) playback is also possible at various speeds, see Figure 3.

[067] View-dependent decoding: We address the problem of encoding data for view-dependent decoding. The process for deciding which cameras are required for the view-dependent decoding of a given rendering image frame is similar to the technique described by Wuermlin et al., i.e., given the viewpoint 401 and the camera calibration data 209, we can determine the contributing cameras and decode and interpolate accordingly.

[068] Compression

[069] Correlation in Image Space

[070] Similar to standard image compression algorithms, 2D transforms, such as wavelet transforms or a discrete cosine transform (DCT) can be applied to encode our input images 202.

[071] However, we are only interested in parts of the images that depict the object 102, as indicated by the masks 201. Therefore, we use a shape-adaptive wavelet encoder.

[072] As shown in Figure 4, the encoder 400 arranges the colors of the relevant pixels linearly by traversing the unmasked portion 402 of each input image 202 in a raster scan order.

[073] Then, we apply a one-dimensional wavelet transform to this list to obtain wavelet coefficients. We do this with a lifting scheme, see Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," Applied and Computational Harmonic Analysis, 3(2):186-200, 1996. The wavelet coefficients are encoded finally by a zerotree coder, and further compressed by arithmetic encoding see Said et al., "A new fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Transactions on Circuits and Systems for Video Technology, 6:243–250, June 1996, Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on Signal Processing, 41:3445–3462, December 1993, and Rissanen et al., Arithmetic coding. IBM Journal of Research and Development, 23(2), pp. 149 - 162, 1979.

[074] The progressive behavior of both the zerotree and arithmetic coder permits a lossy compression up to a desired bit rate or distortion. Furthermore, the encoders enable progressive transmission and partial decompression of the coded bitstream 208 during playback of the video. The decoder 400 is also provided with the lossless silhouette mask 201 of the current frame to be able to reconstruct the scan order used during the encoding.

[075] Correlation in the Time Dimension

[076] We first consider the case of the constrained free viewpoint video. In most cases, the video is played back with increasing time t and at a normal playback speed. Hence, we use the information from previous frames to construct a current frame. A first frame in a segment is a key frame, and

following frames in the segment are encoded as difference frames. Each difference frame indicating a change between the current frame and previous frames.

[077] For each camera i, a decoding function $c_i(t)$ returns a contribution to the 3D point model of the respective camera at time t. If a temporal correlation is exploited by using the information from previous frames, then the decoding function has a form

$$c_i(t) = c_i(t') + \Delta c_i(t) \qquad (1)$$

[078] with $t' < t$, and where $\Delta c_i(t)$ describes the specific contribution of frame t.

[079] Note that most 2D video coding methods are of the form expressed by equation (1). This approach is also feasible for the constrained free viewpoint output video according to the invention.

[080] In the case of the unconstrained free viewpoint video, it is more difficult to exploit temporal correlation. The decoder is required to implement a function f, which returns a 3D point model for any time instant θ during the observation. This implies a viewpoint $v(\theta)$ and a mapping function $m(\theta)$, which maps the rendering time to the recording time.

[081] For a viewpoint v, a weight function $w(v)$ indicates the cameras that contribute to a visible part of the 3D point model. In a first approximation, we can assume that $w(v)$ returns 1 when a camera has a visible contribution, and 0 if not.

[082] We obtain

$$f(v(\theta), t) = w(v(\theta)) \cdot f(t)$$

$$= \begin{bmatrix} w_0(\theta) & \dots & w_{N-1}(\theta) \end{bmatrix} \cdot \begin{bmatrix} c_0(t) \\ \dots \\ c_{N-1}(t) \end{bmatrix}.$$

[083] Assume $\theta' = m^{-1}(t')$ and $t \neq t'$. The decoding of $c_i(t)$ requires the decoding of $c_i(t')$. However, if $w_i(\theta') = 0$, the condition of view-dependent decoding is violated. Hence, optimal view-dependent decoding can only be implemented using decoders defined as

$$c_i(t) = C_i + \Delta c_i(t),$$

[084] with C_i representing a time independent base representation.

[085] Thus, the decoder for the constrained viewpoint video needs to implement decoding in constant time for frames accessed in a *random* order.

[086] Compression Framework

[087] Figure 2 shows the basic system structure 200 for encoding a video according to the invention. The system includes the following encoding modules: a shape encoder 210, a surface normal encoder 220, a position encoder 230, a splat size encoder 240, and a color (texture) encoder 250. The system also includes a geometry construction module 260, and a multiplexer 270.

[088] It should be noted, that the decoder 400 includes a complementary decoding module for each of the encoding modules. The construction of each decoder is self-evident from the construction of the corresponding encoder described in detail herein.

[089] The inputs to the system are segmentation masks 201, camera images 202, and camera calibration data 203. The segmentation masks are obtained from the camera images using any known binary segmentation procedure. The segmentation masks 201 are used to obtain only the foreground (object) pixels to be encoded. Therefore, there is one mask for each camera image. Only the foreground pixels in the camera images 202 define the object 102 in the scene.

[090] We use a lossless encoding for the segmentation masks 201 to avoid shifts and wrong associations between model points and their attributes. The segmentation masks are available to both the encoder and the decoder.

[091] The underlying data representation for our video format is the dynamic 3D point model 265. In the point model, each point has a set of attributes. Because the point attributes are stored and compressed separately, a reference scheme is used. The reference scheme allows for the unique identification between points and their attributes. The camera images 202 are used to build the point model 265. There is an identified sequence of images for each camera. Each point is identified uniquely by its 2D position in image space and an identifier of one or more cameras. We consider only foreground pixels in each image to build the 3D point model.

[092] Pixel attributes can be encoded using a lossy compression scheme. Nevertheless, a lossless or almost lossless decoding is possible when all data are available to the decoder 400.

[093] The bitstream 208 finally consists of encoded key frames and encoded difference frames. The difference frames rely upon a prediction based on a most recent key frame, e.g., frame 0 is the key frame, frame 1 is a difference frame based on frame 0, frame 2 is a difference frame based on frame 1 and frame 0, and so forth.

[094] From the segmentation masks 201 and the camera calibration data 209, a geometric reconstruction of the object of interest in the scene is determined. The output of the geometric reconstruction are surface normal vectors 261, 3D positions 262, and surface splat sizes 263.

[095] The outputs of the encoders are sent to the multiplexer 270. The output of the multiplexer are combined into the encoded video bitstream 208. The camera calibration data 209 need only be sent once to the decoder as long as the cameras 101 remain fixed in place. These parameters can be sent before the start of streaming the video 208. The parameters can be sent using any conventional technique, for example as part of a preamble of the bitstream 208, or on a side channel.

[096] Video Encoding

[097] Constrained Video Encoding

[098] Shapes

[099] The shape encoder 210 can use MPEG-4 lossless binary shape encoding, see Katsaggelos et al., "MPEG-4 and rate/distortion-based shape-coding techniques," Proceedings of the IEEE, 86(6), pp. 1126–1154, June 1998.

[0100] Surface Normals

[0101] The surface normal vectors are progressively encoded using an octahedron subdivision of a unit sphere, see Botsch et al., "Efficient high quality rendering of point sampled geometry," Proceedings of the 13th Eurographics Workshop on Rendering, pp. 53–64, 2002. Two byte codewords are represented in two gray scale MPEG video objects. For most applications, the precision of one byte encoded normals is sufficient.

[0102] Positions

[0103] The x and y coordinates of each point are inherently known from the image pixels and camera calibration data. Therefore, it is sufficient to just encode the depth (z) values. Disparity encoding can be done with MPEG-4 video object coding, where the depth values are quantized as pixel luminance or intensity values.

[0104] Splat Sizes

[0105] The splat sizes are quantized to one byte and the codewords are represented in a gray scale MPEG video object.

[0106] Colors

[0107] The color coding can use conventional MPEG-4 video object coding in the 4:1:1 YUV format, see Ostermann et al., "Coding of arbitrarily shaped video objects in MPEG-4," Proceedings of the International Conference on Image Processing," pp. 496–499, 1997. However, our encoder 200 is also capable of handling texture data in other formats.

[0108] The complete decoding 400 of one constrained video frame requires for each reconstruction view a gray scale MPEG video objects for depth, surface normal, and splat size, and one color video object.

[0109] Unconstrained Video Encoding

[0110] Figure 5 shows an encoder 500 for one attribute of an unconstrained free viewpoint video. In this case, each video 202 is processed as segments 501 of N frames.

[0111] First, a key frame 502 is constructed. In this case, the key frame is an average 510 of the N frames in each segment 501, instead of just the

first frame of the segments as in the constrained case. The average can e.g. be the mean or the median value.

[0112] The key frame is progressively encoded 520.

[0113] Then, a difference 530 is determined between the key frame 501 and each frame of the segment. The difference frames 531 are likewise encoded 520. This produces $N+1$ encoded frames 503. If the value of N is relatively large, for example a hundred or more, then the additional cost of encoding the averaged key frame is marginal for the unconstrained case when compared with the constrained case of N frames.

[0114] As shown in Figure 6, the encoded video bitstream 208 thus comprises a base layer 601 which contains the averaged encoded key frames, and an enhancement layer 602 composed of the encoded difference frames.

[0115] Figure 7A shows a key frame obtained by averaging a segment of a video showing a person turning in a circle. Figures 7B show the reconstruction for one frame of the segment.

[0116] Unconstrained Coding

[0117] Shape Coding

[0118] Shape encoding can be done with lossless JBIG or MPEG-4 binary shape encoding for the segmentation mask.

[0119] Key frame: Those pixels that are foreground pixels in more than half of the frames of the segment 501 are foreground pixels in the key frame 502.

[0120] Difference frame: The difference of the current frame with respect to the key frame.

[0121] Attributes

[0122] All point attributes are encoded using the average encoding scheme 500. That is, the attributes of the key frame in the unconstrained case are obtained from an average of all frames in the segment, while the attributes of the key frame in the constrained case come only from the first frame.

[0123] Key frame: Averaged attributes of the segment.

[0124] Difference frame: Difference with respect to the key frame.

[0125] Now, we describe how the different attributes are represented.

[0126] **Disparity Encoding**

[0127] **Color Encoding:** Embedded zerotree wavelet encoding.

[0128] **Surface Normal Encoding:** The surface normal vectors are progressively encoded using the octahedron subdivision of the unit sphere. The two-byte codewords are then represented in two luminance images which are independently compressed using EZW.

[0129] **Splat Sizes:** The splat sizes are quantized on a gray scale image.

[0130] For large segments 501, the extra cost of decoding the key frames is small. The complexity is about the same as for the unconstrained method. For each reconstruction camera, we have one binary shape image, four gray scale images, and one color image.

[0131] **Multiplexing**

[0132] Because all attributes are encoded independently and progressively, a stream satisfying a given target bit rate can be composed by multiplexing the individual attribute bitstreams into one bitstream for transfer. Appropriate contributions of the single attribute bitstreams are determined according to desired rate-distortion characteristics.

[0133] For example, a bitstream of 300 kilobits per second contains 30 kb/s of shape information, 60 kb/s of position information, 120 kb/s of color information, 45 kb/s of surface normal information and 45 kb/s of splat size information.

[0134] **Extension to Entire Dynamic 3D Scenes**

[0135] The encoders, described so far for video objects, can also be used to encode entire dynamic scenes. Distinct objects in the scene can be encoded in different layers. Static objects are described by a single key frame. A scene graph, which is stored as side information, describes the spatial relations between the different layers. View dependent decoding is again enabled by decoding only those layers which are visible from the current arbitrary viewpoint 401.

[0136] **Dynamic Point Sample Processing and Rendering**

[0137] We perform decoding and rendering in real-time. The compositing 150 combines decoded images with the virtual scene 151 using Z-buffering. We can also provide for post-processing operations, such as 3D visual effects, e.g., warping, explosions and beaming, which are applicable to the real-time 3D video stream, without affecting the consistency of the data structure.

[0138] Furthermore, it is possible to estimate splat sizes and surface normals from the positions of the decoded point samples only. This estimation can be performed in real-time during the rendering process. In

that case, the surface normal encoder 220 and splat size encoder 240 are not required, and storage or transfer of splat sizes and surface normals is unnecessary.

[0139] Point Sample Rendering

[0140] We render the point samples as polygonal splats with a semi-transparent alpha texture using a two-pass process. During the first pass, opaque polygons are rendered for each point sample, followed by visibility splatting. The second pass renders the splat polygons with an alpha texture. The splats are multiplied with the color of the point sample and accumulated in each pixel. A depth test with the Z-buffer from the first pass resolves visibility issues during rasterization.

[0141] View-Dependent Rendering

[0142] To render a particular point, we use a selected set of k decoding active cameras. However, this may produce transition artifacts when the set of active cameras changes during the trajectory of the viewpoint. Therefore, the contribution of each camera to the rendering is weighted by the angle between its viewing direction and the direction of the current viewpoint 401. In order to achieve a smooth transition, we subtract the weight of the closest inactive camera so that only the active cameras have a positive contribution.

[0143] For the current viewpoint, we render k separate images of the scene, each time using only the points from one different active camera.

Finally, the images are combined by blending the point splats using alpha values according to the computed camera weights.

[0144] Although the invention has been described by way of examples of preferred embodiments, it is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.